Realization of Communication between Linux and Windows Based on Socket

Fei Li, Lin Yuan, Zhihuo Wang, Jin Mao
College of Automation, Northwestern Polytechnical University, Xi'an, China, 710072
Email: lifei02468@163.com

Abstract: To realize the remote control, need to get real-time images of the screen, in this paper, using the library function of Glib and GTK+ in GNOME to get the desktop environment of Linux, and then through the Socket Network programming interface to build a suitable Linux and Windows operating system, cross-platform network communication program.

Keywords: Linux; Windows; gtk; Socket

基于 Socket 的 Linux 与 Windows 平台间通信的实现

李　菲，袁　琳，王志伙，毛　晋
西北工业大学自动化学院，西安，中国，710072
Email: lifei02468@163.con

摘　要：为实现远程控制，需要获得计算机屏幕的实时图像，本文利用 GNOME 的 Glib 库和 GTK+ 库来获取 Linux 桌面屏幕，然后通过 Socket 网络编程接口，构建出一个适合于 Linux 和 Windows 操作系统的、跨平台的网络通信程序。

关键词：Linux；Windows；gtk；Socket

1 引言

目前，Linux 操作系统由于其系统源码的公开性和系统性能的稳定性，被广泛用做服务器操作系统。而 Windows 由于其友好的图形界面，强大的编程环境得到了最广泛的应用，特别是在 PC 机上更是独霸市场。这就使得很多部门单位在使用中出现了两大操作系统共存的情况，从而也就出现了如何在这两大操作系统间进行实时通信的问题。

本文在掌握了 Linux 与 Windows 下的应用程序开发技巧后，实现了 Linux 屏幕的抓取，并将抓取的桌面图片通过套接字编程传输到 Windows 平台之上，成功实现了 Linux 和 Windows 平台下实现点对点的文件传输，基本解决了这两大平台的通信问题。

2 Socket 编程

在基于 TCP/ IP 协议的网络中，Socket[1] 是网络通信的基本操作单元。它提供了不同主机进程双向通信的端口，这些进程在通信前各自建立一个 Socket，并通过对 Socket 的读/写操作来实现网络通信。基于 TCP/ IP 协议的 Socket 编程是一种典型的会话编程方式，它既适用于客户/服务器通信方式，又适用于点对点通信方式。

一旦应用程序创建了一个 Socket，并进行地址绑定与外部地址 TCP 连接，就可以利用套接字描述字作为参数使用 Write 在此连接上发送数据流，在连接的另一端则使用 Read 接收数据，这样 Socket 就连接了两端的应用程序。套接字机制提供了一系列的系统调用函数，它们都是围绕着应用程序如何利用网络通信协议在网络上进行数据交换而设计的。通过这些函数的调用，应用程序就可以在掩盖通信协议细节的情况，实现网络传输。

3 通信环境的建立

该程序是一个跨平台的通信程序，服务器运行 Linux 操作系统，客户端运行 Windows 操作系统。在本应用中，采用的是虚拟机上运行 Linux 操作系统，作为服务器接收客户端命令，运行 Linux 的屏幕抓取程序。而客户端是 Windows xp 操作系统，向服务器

978-1-935068-40-2

发出请求，请求其进行屏幕抓取然后再回传给客户端，完成一次屏幕的抓取和传输。

在着手进行抓屏和传输之前必须要保证服务器和客户端之间可以进行正常的网络通信。虚拟机设置成桥接网络连接模式， 采用这种网络连接模式后，对应虚拟机就被当成主机所在以太网上的一个独立的物理机来看待。虚拟机和主机设置在同一个网段，网关、DNS 也都与主机设置成一样。

虚拟机配置好以后，要先检测一下主机和虚拟机是否能正常通信。通过 ping 命令来检测。如果 ping 成功则可以运行相应的程序进行通信，否则得检查各种情况使其可以正常通信。

服务器端程序的开发和测试都是在 Linux 环境下进行的，其中包含大量的 Linux 库文件，如果移植到其他操作系统中，可能无法运行。在传输文件的程序中，服务器与客户端采用的是面向连接的可靠的 TCP 连接。服务器处于监听状态，当有客户要求服务时，服务器端进行阻塞，来处理客户端的请求。服务器端允许自由设置自己最大的连接客户数。

4 屏幕抓取

Linux 屏幕抓取与传输技术的实现可分为两个大的模块，一个是抓屏程序,另一个是网络传输，而网络传输又分为服务器端（Linux）和客户端（Windows）。

下面介绍程序的各个模块，为了方便，在程序流程图中的开始阶段标出了每个模块的主函数名称，而在流程图的每个步骤也把主要用到的函数名列在框图里。

4.1 抓屏模块

本程序的核心模块是 Linux 的屏幕抓取[2]，图 1 是抓屏模块的流程图。该模块的工作过程如下：

(1)将当前屏幕设为默认的需要抓取的屏幕。

(2)要抓取屏幕，就需要获得当前屏幕像素值。

(3)要将抓取的屏幕像素值放到一个窗口中，所以要获得一个可以显示屏幕的根视窗。还要指出该视窗的起始坐标。

(4)给定一个可绘图区，从而可以将（3）确定的窗口放入到该区域。

(5)将该窗口内容保存到 pixbuf 的缓存中，该内容被保存成 JPEG 格式以便传送到客户端。

(6)释放内存。

Figure 1. Flowchart: Screen capture module

图 1. 抓屏模块的流程图

4.2 参数设置

该抓屏程序主要是利用 GNOME 的 Glib 库和 GTK+库获取 Linux 桌面屏幕的方法。该程序支持命令行模式，用户可以指定抓取的屏幕文件的名称以及指定抓屏的时间间隔，如果用户没有指定的话，就进入默认模式。

(1)localtime(&tt);//获取抓屏时刻时间。

(2)strftime(file,32,"%Y%m%d%H%M%S", tm_ptr)；//将文件名设为抓屏时间。

(3)strcat(file, ".jpg")；//文件保存的格式是 jpg。

可以通过 -o 【文件名】指定要保存的文件名称。

可以通过 -s 【时间间隔】指定要隔多长时间允许程序运行抓屏程序，其实是一个定时程序。

5 图片传输

图片传输是基于 TCP/IP 协议的，该协议允许创建

和维护与远程计算机的连接，使其彼此可以进行数据传输。利用 TCP/ IP 协议通讯必须分别建立客户应用程序和服务器应用程序。在利用 Socket 编程时，一般工作模式是客户进程向服务器进程发出请求，服务器进程执行被请求的任务并将结果返回给客户进程。

下面分别介绍在 Linux 平台的服务器程序和在 Windows 平台的客户程序。

5.1 服务器端模块

服务器端屏幕抓取程序的实现过程如图 2。

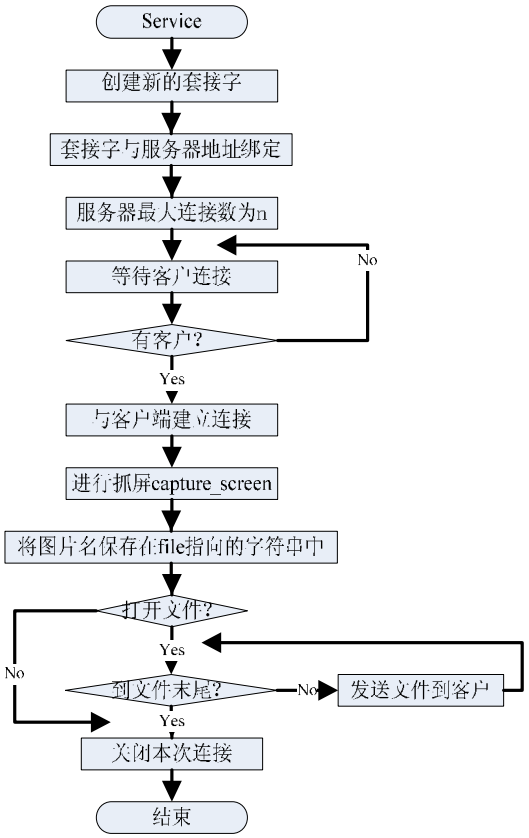

Figure 2. Flowchart: Server process

图 2. 服务器程序流程图

在Linux中通过结构sockaddr_in 来定义套接字通信的基本信息,然后通过该结构由系统分配套接字。具体实施过程包括以下三个步骤：

(1)定义套接字变量

```
struct sockaddr_in server; //服务器通信地址
struct sockaddr_in client；//本地通信地址信息
int     listensd, connectsd；//套接字句柄变量
```

(2)初始化相关信息,并向系统请求分配套接字。

```
if((listensd=socket(AF_INET,SOCK_STREAM,0))==
-1 ) { //分配服务器端用于监听的套接字
                perror("socket");
                  exit(1); }
      setsockopt(listensd,SOL_SOCKET,
                SO_REUSEADDR, &opt,sizeof(opt));
//设置监听的套接字选项
bzero(&server,sizeof(server));
server.sin_family = AF_INET; //所用协议是TCP
server.sin_port = htons(SER_PORT);
//端口号转成网络字节顺序
server.sin_addr.s_addr = htonl(INADDR_ANY);
//将IP地址转换成为内部表示的地址信息
if(bind(listensd,(struct  sockaddr*)&server,sizeof(server))
< 0) {//将该套接字与服务器通信地址信息绑定
            perror("Bind");
            close(listensd);
            exit(1); }
(3)监听客户端连接请求,并等待客户连接请求。
if( listen(listensd,5) == -1 ) {//建立长度为5的请求队列
             perror("listen");
             close(listensd);
             exit(1); }
while( 1 ) {int     sin_len = sizeof(client);
      if((connectsd=accept(listensd,(struct
      sockaddr *)&client,&sin_len)) < 0 )
//监听客户连接请求,没客户端连接,程序将一直等待
                {perror("accept");
                 continue; }
                  capture_screen(file, sec);
//连接后服务器抓屏，  进入默认模式
      if((fp=fopen(file,"rb"))==NULL){
                perror("File   open");
                close(listensd);
                exit(1); }
      while(!feof(fp)){len=fread(buf,1,MAX_LEN,fp);
          if(len!=send(connectsd,buf,len,0)) {
                perror("write");
      //向客户端发送文件信息，直到读到文件末尾
                          break; }     }
                             close(connectsd);
                             fclose(fp);}
          close(listensd);
               return     0; }
```

5.2 客户端模块

客户端运行该程序就可与服务器建立连接，接收服务器端所抓取的 Linux 屏幕。具体流程图如图 3 所示。

(1)初始化相关信息,并向系统请求分配套接字。

因为客户端运行的是 Windows 系统，所以与服务器在这一步骤不尽相同。

978-1-935068-40-2 © 2010 SciRes.

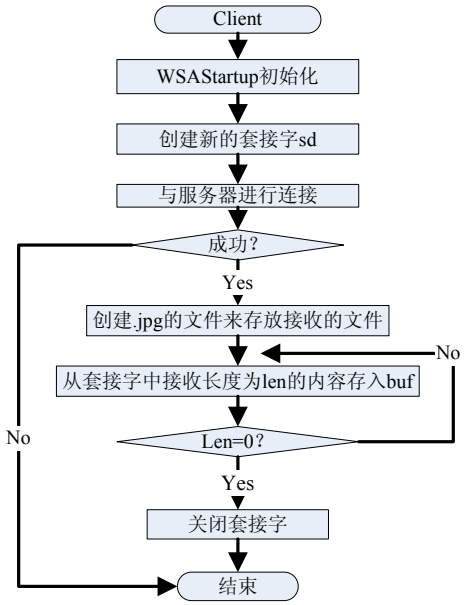

Figure 3. Flowchart: Client process
图3. 客户端程序流程图

WORD wVersionRequested[4, 5];//指明 Socket 版本
WSADATA wsaData; // Windows Sockets 实现细节
int err;
wVersionRequested = MAKEWORD(2, 2);
err = WSAStartup(wVersionRequested, &wsaData);
if (err != 0) { return 0; }

/*本函数必须是应用程序或 DLL 调用的第一个 Windows Sockets 函数.它允许应用程序或 DLL 指明 Windows Sockets API 的版本号及获得特定 Windows Sockets 实现的细节.应用程序或DLL 只能在一次成功的 WSAStartup()调用之后才能调用进一步的 Windows Sockets API 函数.*/

sd=socket(AF_INET,SOCK_STREAM,0) ;
server.sin_family = AF_INET;
server.sin_port = htons(SER_PORT);
server.sin_addr.s_addr=inet_addr("20.22.110.99");
//Linux 服务器地址

(2)向服务器端发出连接请求。

connect(sd,(struct sockaddr_in*)&server,sizeof(server));

客户端要特别注意指定的 server.sin_port 要与服务器端的端口号相同。在本程序中服务器端口号为 1900。

6 结束语

本文研究了 Linux 的屏幕抓取以及 Linux 与 Windows 的网络传输的设计与实现，双方采用 TCP/IP 协议，运用 socket 接口来开发网络通信程序。套接字使得连接在通信两端的进程，能够通过各自的套接字发送和接收消息，从而实现了 Linux 与 Windows 平台间的通信。

致 谢

在此真诚地感谢所有帮助、关心和支持我的老师、同学、朋友和家人！祝愿他们身体健康，天天开心！

References (参考文献)

[1] Feng Li. Implementation of Socket_ Stream Communication Program in Windows and Linux[J].Microcomputer Information, 2006. 22(3): 112-113,75.
李峰. 利用流式 Socket 编程实现 Windows 与 Linux 的通信[J]. 微计算机信息, 2006. 22(3): 112-113,75.

[2] Hong Luo, Dejun Mu. Compression and Transmission Scheme of Linux Desktop Graph Sequences[J]. Computer Engineering, 2006. 32(7): 221-223.
罗红,慕德俊. Linux 桌面图形序列的压缩与传输[J]. 计算机工程, 2006. 32(7): 221-223.

[3] Jon Masters, Richard Blum. Professional Linux Programming [M]. 2007, Wrox. 432.

[4] Kun Zhou, Desheng Fu. Data transmission and security on net based on Windows Socket[J]. Computer Engineering and Design, 2007. 28(22): 5381-5383.
周坤,傅德胜. 基于 Windows Socket 的网络数据传输及其安全[J]. 计算机工程与设计, 2007. 28(22): 5381-5383.

[5] Rong Tang. Data conversion under multi-communication——the achievement of data communication between GPS locating mobile phone and the control center[D],2009,Chengdu University of Technology.17-20.
唐蓉. 多通信方式下的数据转换——GPS 定位手机与总控中心之间数据通信的实现[D], 2009, 成都理工大学. 17-20.